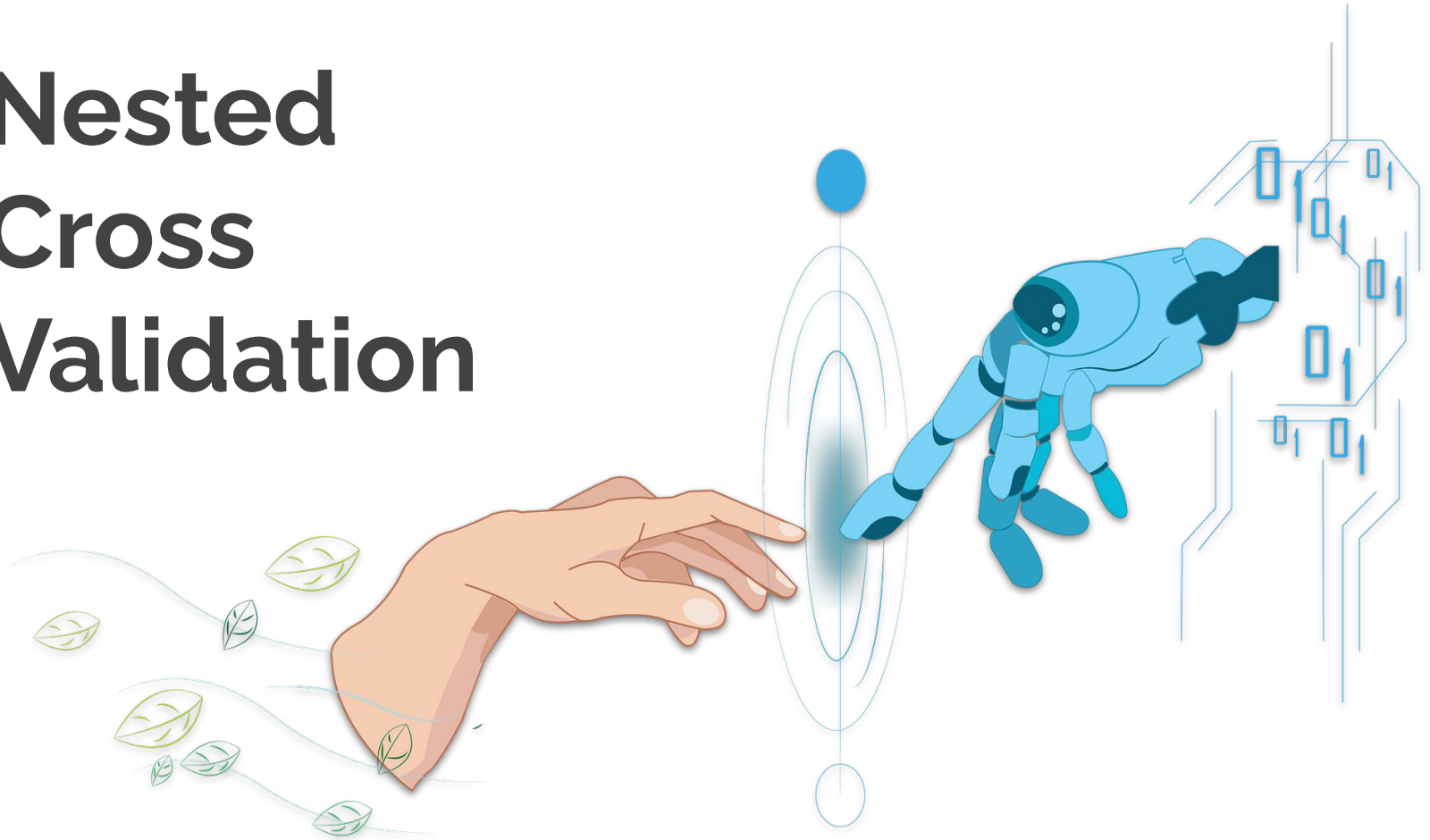


Nested Cross Validation



Why we need k-fold cross-validation

We commonly **evaluate** machine learning models on a dataset using **k-fold cross-validation**.

- Each machine learning model includes 1+

hyperparameters

- allow the model to be customized to a dataset.

Problem

- Rarely are there good heuristics on **how to configure the algorithm hyperparameters** for a dataset

Solution

- Use an **optimization algorithm**
 - discovers the hyperparameters that perform well or **best** on the dataset

Common Optimization Algorithms

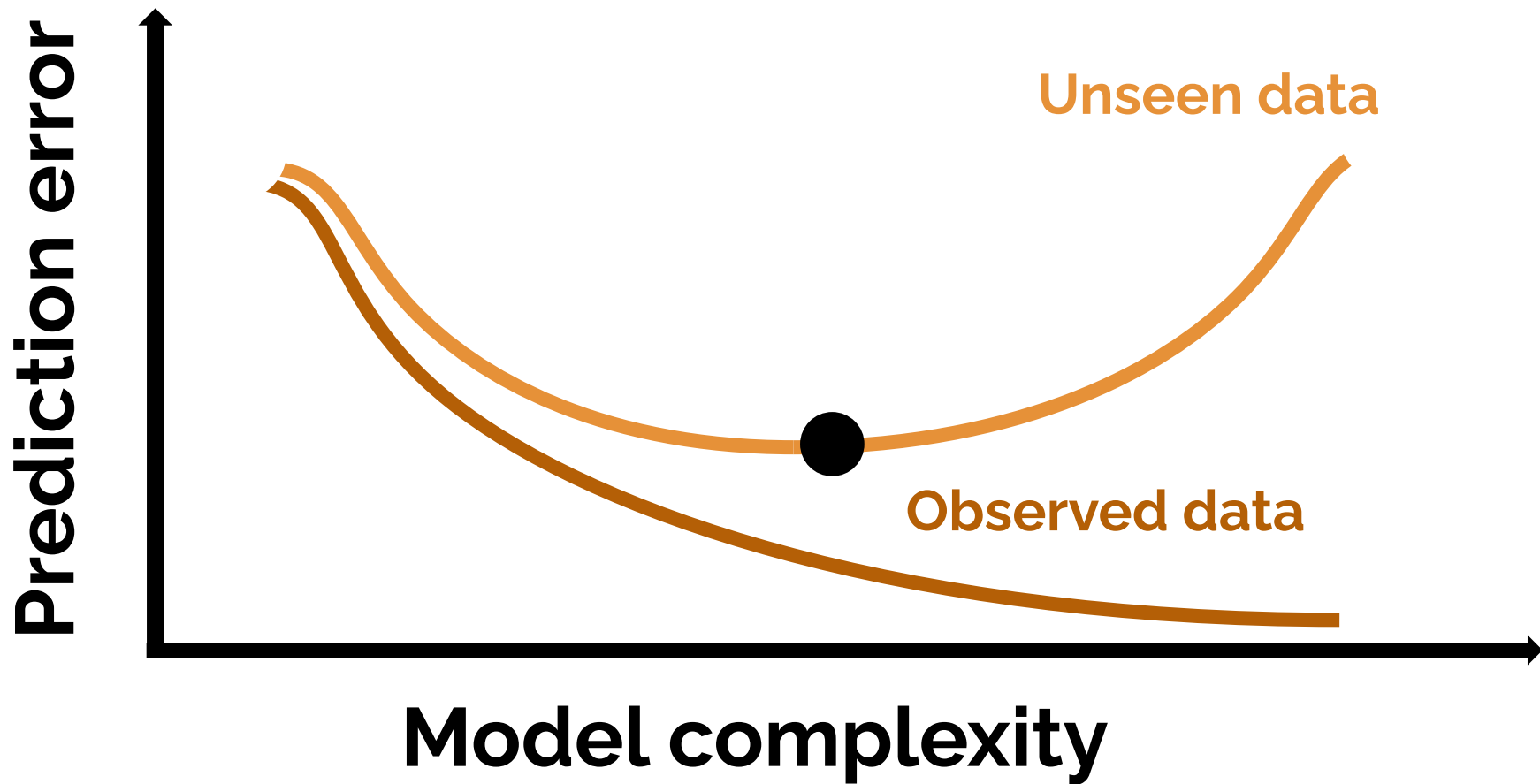
- grid search
- random search

How are the model hyperparameters evaluated?

- Typically with **k-fold cross-validation**
 - More on that later

Central challenge in machine learning

- Algorithm must perform well on new inputs, not just training data



2 Goals of Machine Learning

- **Model selection**
- **Model assessment**

How do you choose a model? 🤔

- Find a model that fits future data best

How do you evaluate a model? 🤖

- Simplest approach: holdout cross-validation

- **What is cross validation?**

Holdout cross-validation

- Randomly split data into:
 - Training set
 - Testing set

NEVER use test set to train

Holdout cross-validation 👎

- Fails to use all available data
- If $\frac{1}{2}$ of data is for training, poor hypothesis
- If $\frac{1}{10}$ of data is for testing, poor estimate of actual accuracy

What is the solution?

- **Cross validation**

- Repeat training and validation on different randomly chosen subsets of original training set

K Fold cross validation

- **Each sample will be used for validation in 1 of the sets**
- **Each sample serves double duty: as training set and validation set**

k-fold cross-validation

- used to estimate the performance of machine learning models **when making predictions on data** *not used during training*.

k-fold cross-validation

- Used when **optimizing** the hyperparameters of a model
- And when comparing and **selecting** a model

Is it good?

- **effective** at estimating an algorithm's performance

Problem

- if it is used multiple times with the same algorithm, it can lead to **overfitting**

Why?

- Each time a model with different hyperparameters is evaluated on a dataset, it provides **information about the dataset**
 - an *often noisy* model performance score

Performance Score

- can be exploited in the model configuration process to find the **best performing model configuration** for the dataset

The Attempt

- k-fold cross-validation attempts to *reduce* this **noisy effect**
 - cannot be removed completely
 - some overfitting of the model hyperparameters to the dataset will occur

**This is the normal case for
hyperparameter
optimization**

Problem

- if *the score alone* is used to **select** a model
- Or if the same dataset is used to **evaluate** the tuned models
 - Model selection will be biased by the **overfitting**

Problem

When the *same* **cross-validation procedure and dataset** tune
and select a model...

- Likely leads to an optimistically **biased** evaluation of the model performance.

- **How do we overcome
this bias?**

Solution

- **nest** the hyperparameter optimization procedure
under the model selection procedure

= **double cross-validation (nested cross-validation)**

nested cross-validation

- **preferred** way to evaluate and compare tuned machine learning models

nested cross-validation

- attempts to solve the overfitting problem
of the training dataset

2 cross-validation loops

- k-fold cross-validation for **model hyperparameter optimization** is **nested** inside the k-fold cross-validation for **model selection**

Why it's a solution

- **hyperparameter search** doesn't have an opportunity to overfit
 - only exposed to a **subset** of the dataset provided by the outer cross-validation procedure

Cost

- large **increase** in the **number** of model evaluations

Nested Cross Validation

